STAT 5291 project

2023-04-21

Transform the daily data to monthly data

```r
# Load required libraries
library(xts)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(zoo)
library(readr)

air_poll_data <- read.csv("air_pollution_ts.csv")

# Ensure the date column is in the Date class format
air_poll_data <- air_poll_data[!is.na(air_poll_data$date), ]
```

```r
air_poll_data$date <- as.Date(air_poll_data$date, format = "%m/%d/%y")

air_poll_xts <- xts(air_poll_data[, -1], order.by = air_poll_data$date)

air_poll_monthly <- apply.monthly(air_poll_xts, FUN = mean)

air_poll_monthly_df <- data.frame(Date = index(air_poll_monthly), coredata(air_poll_monthly))

head(air_poll_monthly_df)
```

```
##         Date coredata.air_poll_monthly.
## 1 2010-01-31                   87.14444
## 2 2010-02-28                   98.26414
## 3 2010-03-31                   98.88642
## 4 2010-04-30                   79.88472
## 5 2010-05-31                   86.91062
## 6 2010-06-30                  110.71319
```

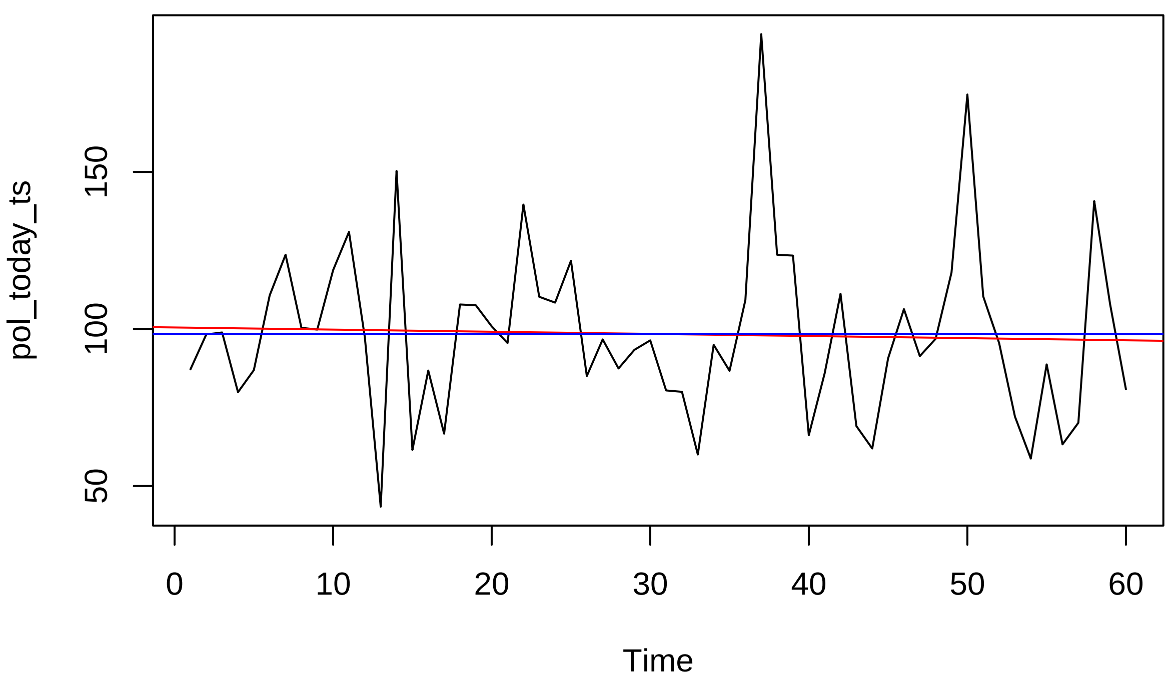
```
#write.csv(air_poll_monthly_df, file = "air_poll_monthly.csv", row.names = FALSE)
```

Data inspection

```
pol_data <- read.table('air_poll_monthly.csv',sep=',',skip=1)
pol_today_ts <- ts(pol_data$V2)
total_length <- length(pol_today_ts)

ts.plot(pol_today_ts)

fit <- lm(pol_today_ts ~ as.numeric(1:total_length))
abline(fit, col = "red")
abline(h = mean(pol_today_ts), col = "blue")
```

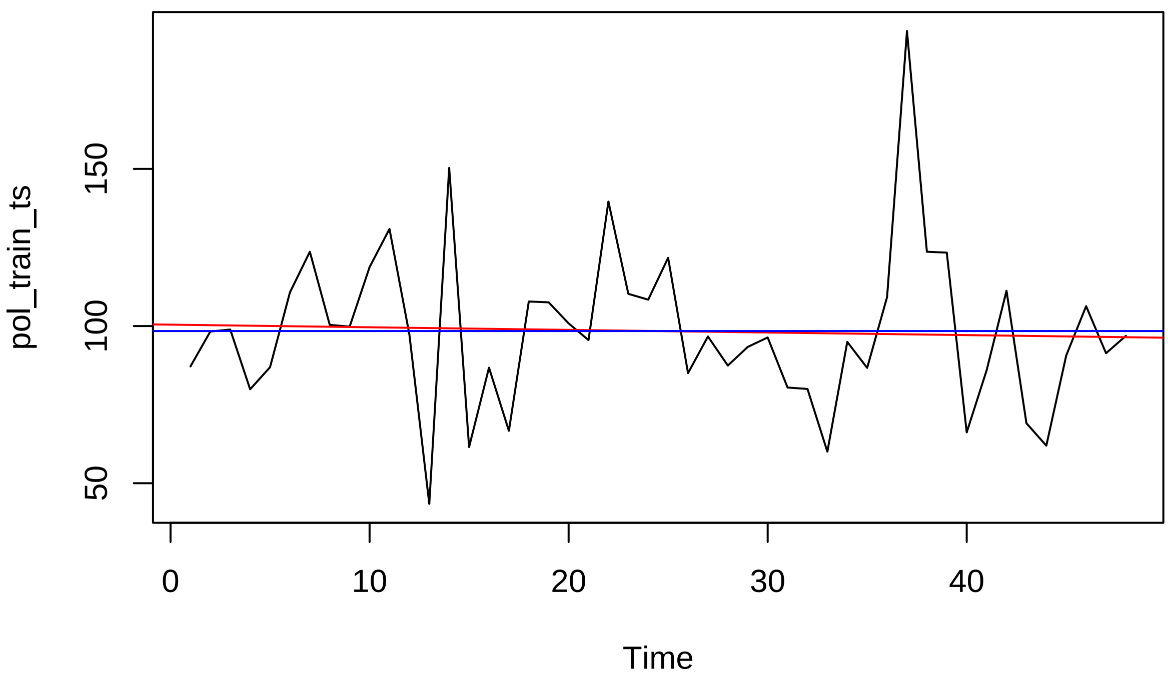



```
# Split the data into training and test sets
train_length <- total_length - 12
pol_train_ts <- pol_today_ts[1:train_length]
pol_test_ts <- pol_today_ts[(train_length + 1):total_length]

ts.plot(pol_train_ts)

fit <- lm(pol_train_ts ~ as.numeric(1:length(pol_train_ts)))
```

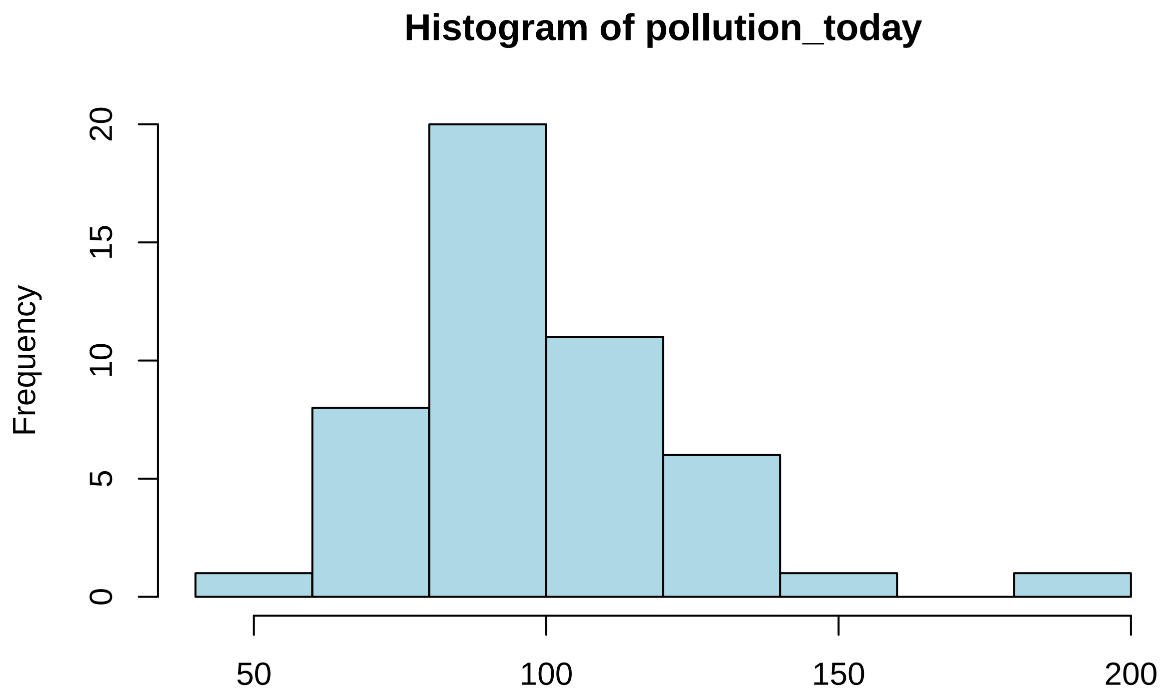
```r
abline(fit, col = "red")
abline(h = mean(pol_train_ts), col = "blue")
```



The red line in the plot represents the linear regression line, which is the best-fitting straight line through the data points. The blue line represents the mean of the time series. As both lines overlap in the plot, it implies that the mean of the time series and the linear regression line are very close or nearly the same.

It suggests that the linear trend in the time series data is essentially flat. This means that, on average, the pollution_today levels have not changed significantly over time. However, a linear regression might not always be the best model to capture the underlying trend in the time series data, especially if the data has a more complex, non-linear pattern. Thus, we are going to try other models.

```r
hist(pol_train_ts, col='light blue', xlab='', main='Histogram of pollution_today')
```

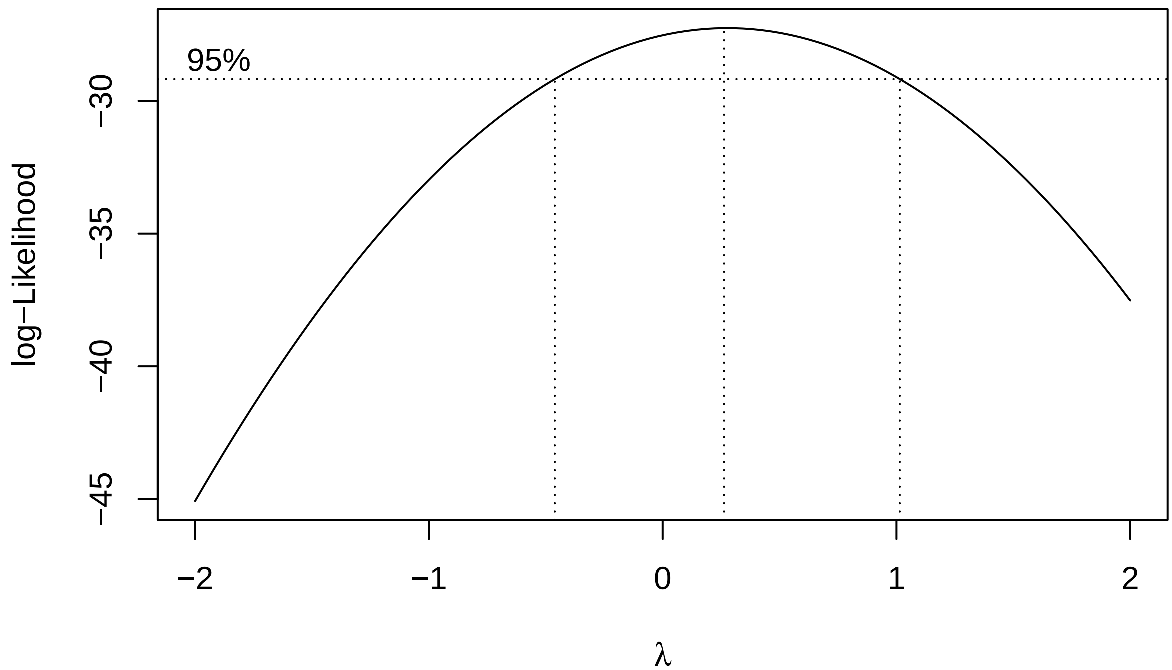


From the histogram of the data, we can tell the data is skewed, which means the variance is non-constant. In order to have constant variance for further steps, we can try Box-Cox transformation.

```
require(MASS)
```

```
## Loading required package: MASS
```

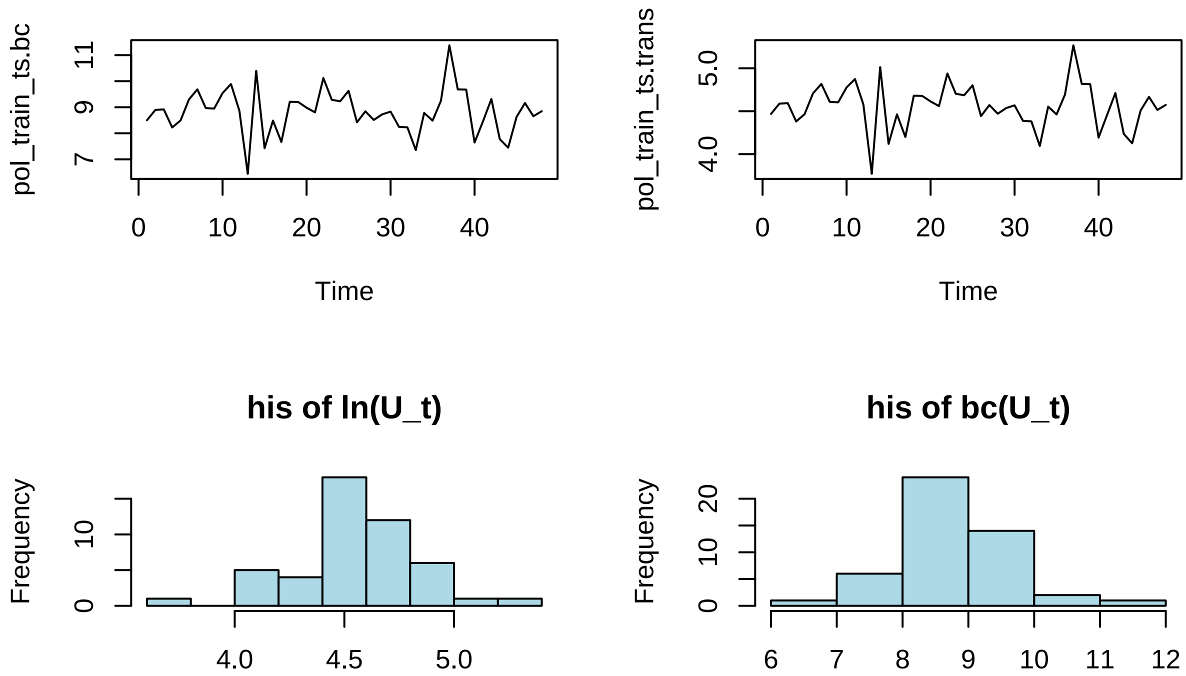
```
bcTrans <- boxcox(pol_train_ts~as.numeric(1:length(pol_train_ts)))
```




```
bcTrans$x[which(bcTrans$y == max(bcTrans$y))]
```

```
## [1] 0.2626263
```

```
lambda<-bcTrans$x[which(bcTrans$y == max(bcTrans$y))]
pol_train_ts.bc = (1/lambda)*(pol_train_ts^lambda-1)
pol_train_ts.trans <- log(pol_train_ts)
par(mfrow=c(2,2))
plot.ts(pol_train_ts.bc)
plot.ts(pol_train_ts.trans)
hist(pol_train_ts.trans,col='light blue',xlab='', main='his of ln(U_t)')
hist(pol_train_ts.bc,col='light blue',xlab='', main='his of bc(U_t)')
```



From the result, $\lambda = 0.26$, but since 0 is also in the confidence interval, for simplicity, we will just use $\lambda = 0$, which is the log transformation.

```
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

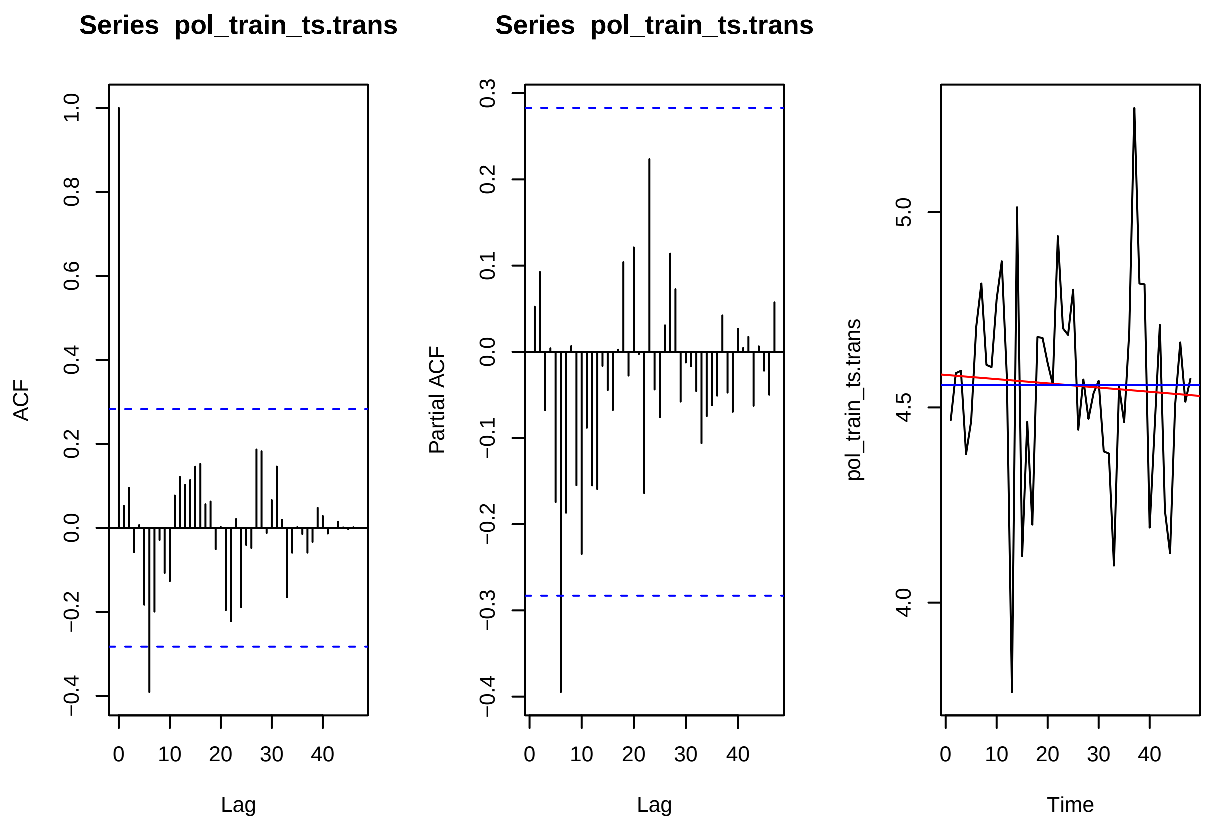
```
adf_result <- adf.test(pol_train_ts.trans)
print(adf_result)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  pol_train_ts.trans
## Dickey-Fuller = -3.1204, Lag order = 3, p-value = 0.1267
## alternative hypothesis: stationary
```

From the test, we can tell our data is stationary, thus can be used for time series analysis.

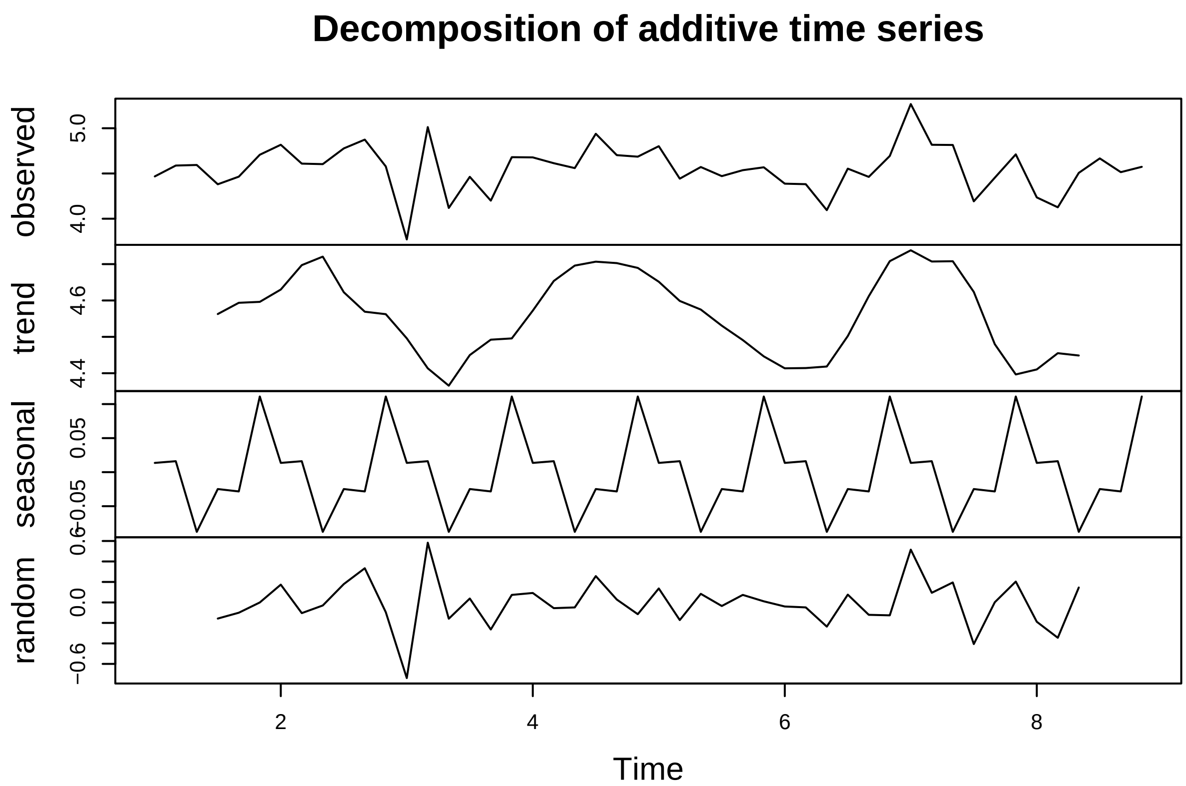
```
par(mfrow=c(1,3))
acf(pol_train_ts.trans, lag.max=50)
pacf(pol_train_ts.trans, lag.max = 50)
```

```
ts.plot(pol_train_ts.trans)
fit <- lm(pol_train_ts.trans~as.numeric(1:length(pol_train_ts.trans)))
abline(fit, col='red')
abline(h=mean(pol_train_ts.trans), col='blue')
```



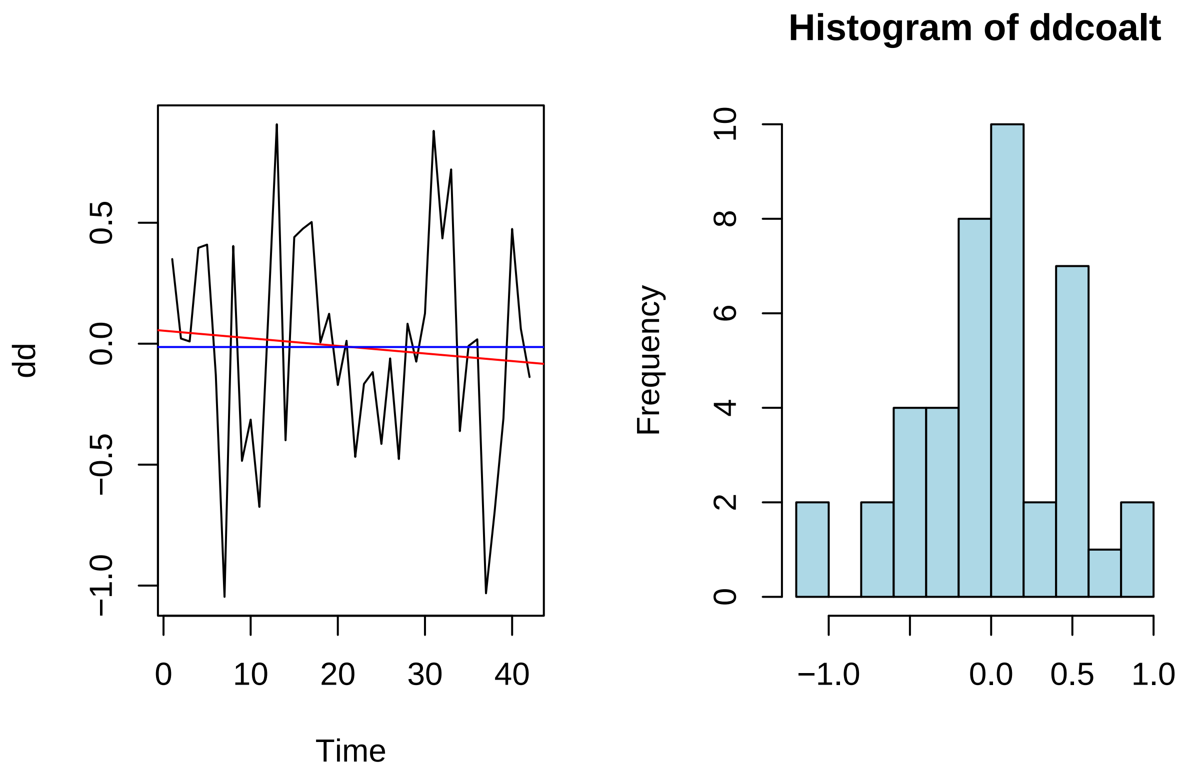

```
library(ggplot2)
library(ggfortify)

y <- ts(as.ts(pol_train_ts.trans), frequency=6)
decomp <- decompose(y)
plot(decomp)
```

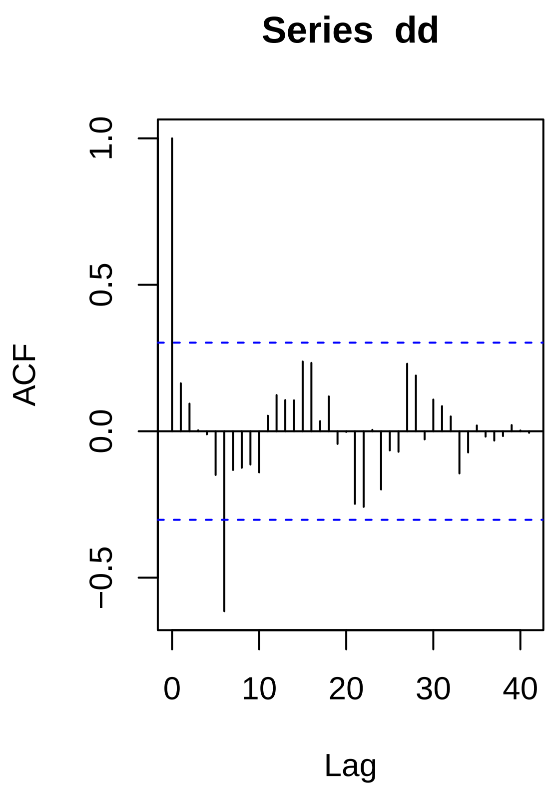


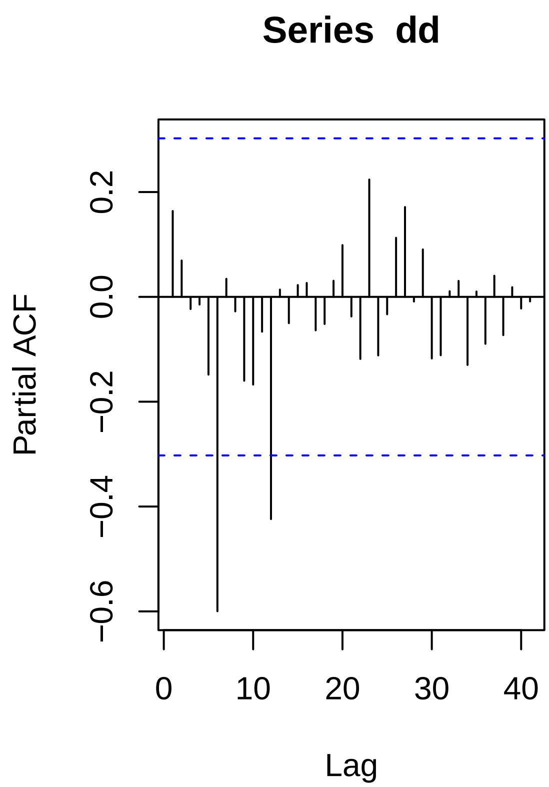
From the decomposition, it shows that our data exists seasonality. From the ACF graph, there is one line at lag 6 that is noticeable, which indicates the seasonality component is 6, so next step is to difference the data once at lag 6 to remove the seasonality.

```
par(mfrow=c(1,2))
dd <- diff(pol_train_ts.trans,lag=6, differences=1)
ts.plot(dd)
fit <- lm(dd~as.numeric(1:length(dd)))
abline(fit, col='red')
abline(h=mean(dd), col='blue')
hist(dd, col='light blue', xlab='', main='Histogram of ddcoalt')
```




```
par(mfrow=c(1,2))
acf(dd, lag.max=50)
pacf(dd, lag.max=50)
```





Try possible models

```
library(qpcR)
```

```
## Loading required package: minpack.lm

## Loading required package: rgl

## Loading required package: robustbase

## Loading required package: Matrix
```

```
library(forecast)
```

```
## Registered S3 methods overwritten by 'forecast':
##   method                 from
##   autoplot.Arima         ggfortify
##   autoplot.acf           ggfortify
##   autoplot.ar            ggfortify
##   autoplot.bats          ggfortify
##   autoplot.decomposed.ts ggfortify
```

```
##   autoplot.ets          ggfortify
##   autoplot.forecast     ggfortify
##   autoplot.stl          ggfortify
##   autoplot.ts           ggfortify
##   fitted.ar             ggfortify
##   fortify.ts            ggfortify
##   residuals.ar          ggfortify
```

```
df <- expand.grid(p=0:2, q=0:2, P=0:2, Q=0:1)
df <- cbind(df, AICc=NA)
for (i in 1:nrow(df)) {
  arima.obj <- NULL
  try(arima.obj <- arima(pol_train_ts.trans, order=c(df$p[i], 0, df$q[i]),
                         seasonal=list(order=c(df$P[i], 1, df$Q[i]), period=6),
                         method="ML"))
if (!is.null(arima.obj)) { df$AICc[i] <- AICc(arima.obj) }
}
```

```
df <- df[order(df$AICc, decreasing = FALSE), ]
head(df)
```

```
##    p q P Q     AICc
## 19 0 0 2 0 21.13268
## 37 0 0 1 1 21.69124
## 20 1 0 2 0 22.71904
## 22 0 1 2 0 22.79455
## 46 0 0 2 1 23.01427
## 38 1 0 1 1 23.50245
```

For the model choosing, we ran a for loop to estimate which model produce the lowest AICc value. I chose models that have the lowest and the second lowest to estimate the coefficients.

```
# Model A SARIMA(0,0,0)(2,1,0)_6
Model_A <- arima(pol_train_ts.trans, order=c(0,0,0),
                         seasonal=list(order=c(2,1,0), period=6),
                         method="ML")
Model_A
```

```
##
## Call:
## arima(x = pol_train_ts.trans, order = c(0, 0, 0), seasonal = list(order = c(2,
##     1, 0), period = 6), method = "ML")
##
## Coefficients:
##          sar1     sar2
##       -1.0719  -0.5924
## s.e.   0.1248   0.1395
##
## sigma^2 estimated as 0.06764:  log likelihood = -7.43,  aic = 20.87
```

```r
AICc(Model_A)
```

```
## [1] 21.13268
```

```r
# Model B SARIMA(0,0,0)(1,1,1)_6
Model_B <- arima(pol_train_ts.trans, order=c(0,0,0),
                         seasonal=list(order=c(1,1,1), period=6),
                         method="ML")
Model_B
```

```
##
## Call:
## arima(x = pol_train_ts.trans, order = c(0, 0, 0), seasonal = list(order = c(1,
##     1, 1), period = 6), method = "ML")
##
## Coefficients:
##          sar1     sma1
##       -0.4098  -0.8164
## s.e.   0.1798   0.3489
##
## sigma^2 estimated as 0.06529:  log likelihood = -7.71,  aic = 21.42
```

```r
AICc(Model_B)
```

```
## [1] 21.69124
```

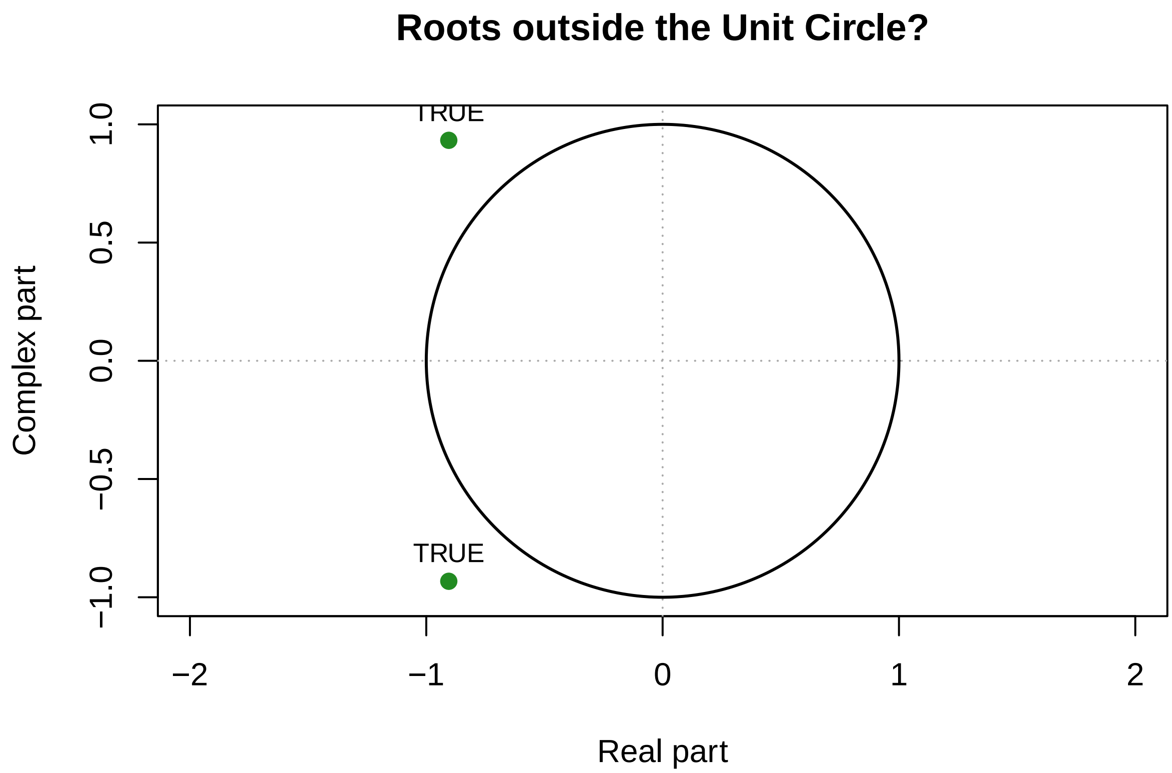
From these, we get two models:

Model A: $SARIMA(0,0,0)(2,1,0)_6$ Model B: $SARIMA(0,0,0)(1,1,1)_6$

Check invertible and stationary

```r
library(UnitCircle)
# Model A
uc.check(pol_=c(1, 1.0719,0.5924), plot_output=T)
```

```
##       real   complex outside
## 1 -0.90471  0.932496    TRUE
## 2 -0.90471 -0.932496    TRUE
## *Results are rounded to 6 digits.
```

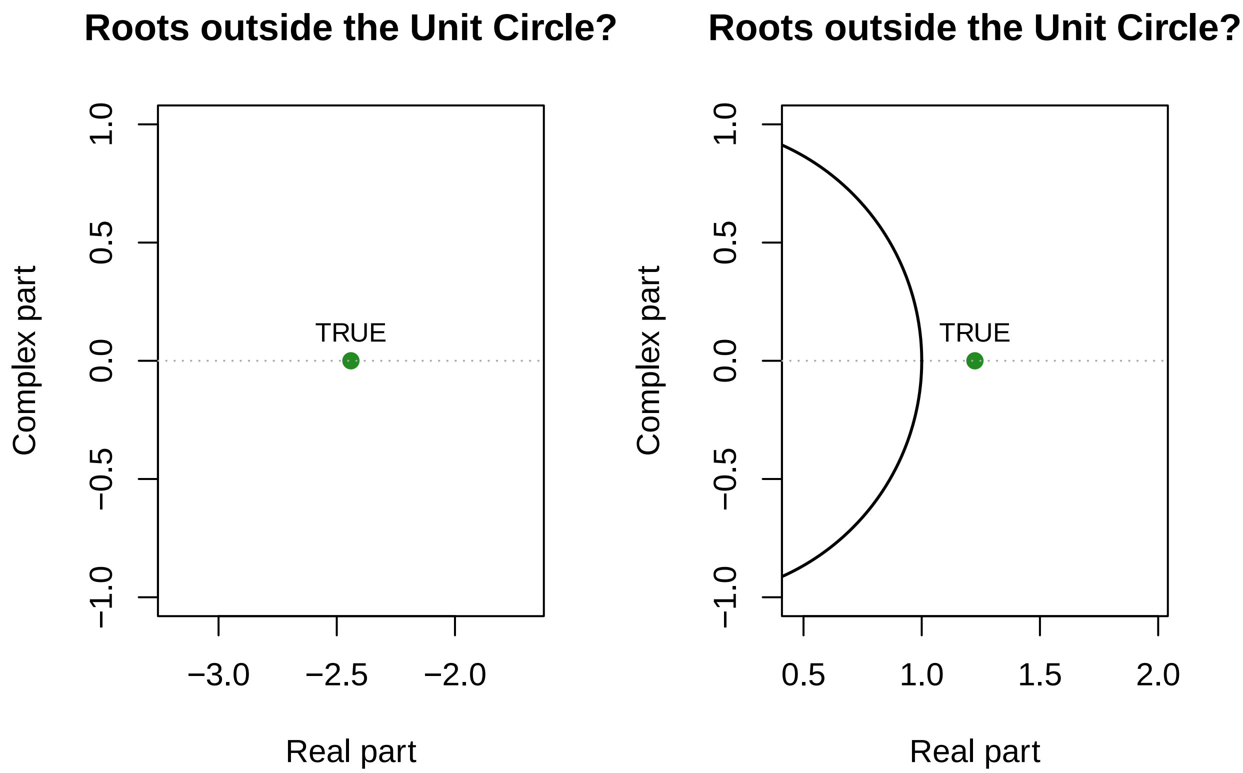



```
# Model B
par(mfrow=c(1,2))
uc.check(pol_=c(1, 0.4098), plot_output=T)
```

```
##        real complex outside
## 1 -2.440215       0    TRUE
## *Results are rounded to 6 digits.
```

```
uc.check(pol_=c(1, -0.8164), plot_output=T)
```

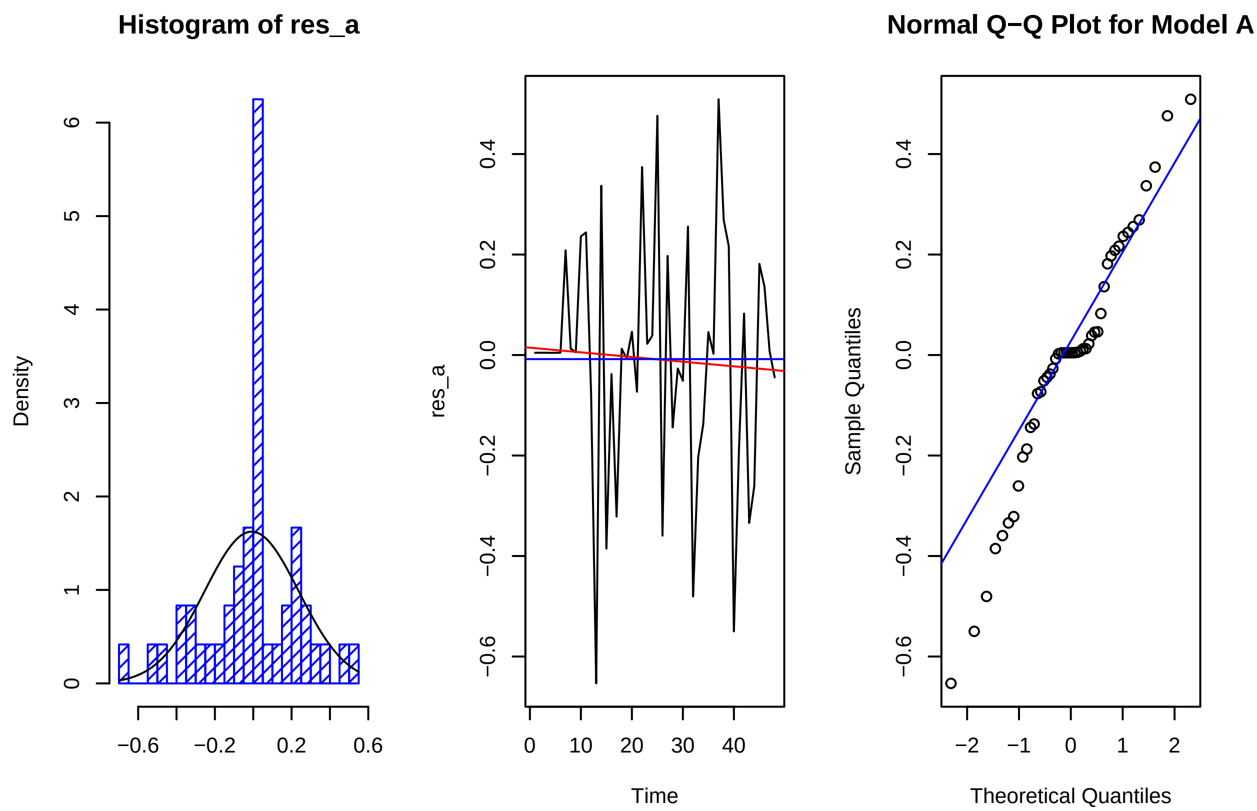
```
##      real complex outside
## 1 1.22489       0    TRUE
## *Results are rounded to 6 digits.
```



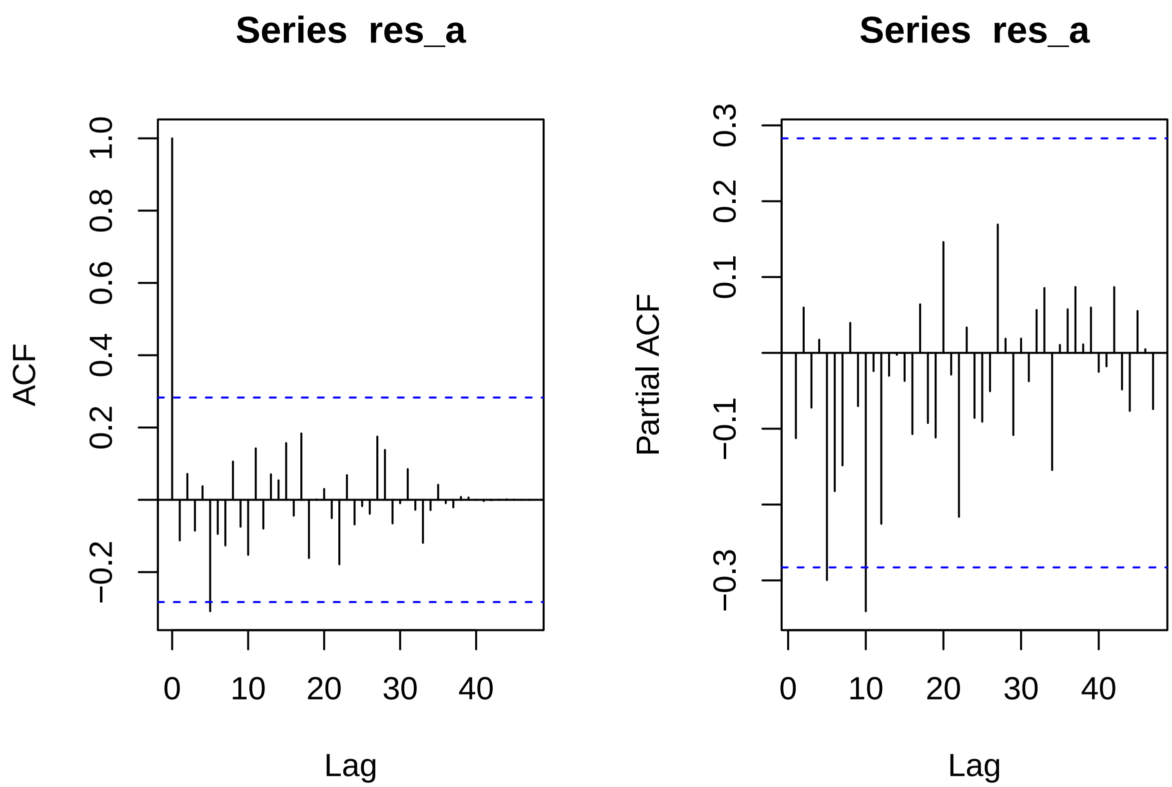
Both Model A and Model B are stationary and invertible.

Diagnostic checking for Model A

```
#Model A
par(mfrow=c(1,3))
res_a <- residuals(Model_A)
hist(res_a,density=20, breaks=20, col='blue', xlab='', prob=TRUE)
m_a <- mean(res_a)
std_a <- sqrt(var(res_a))
curve(dnorm(x,m_a,std_a), add=T)
plot.ts(res_a)
fitt_a <- lm(res_a~as.numeric(1:length(res_a)))
abline(fitt_a, col='red')
abline(h=mean(res_a), col='blue')
qqnorm(res_a, main="Normal Q-Q Plot for Model A")
qqline(res_a, col='blue')
```




```
par(mfrow=c(1,2))
acf(res_a, lag.max=50)
pacf(res_a, lag.max=50)
```




```
shapiro.test(res_a)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res_a
## W = 0.96092, p-value = 0.11
```

```
Box.test(res_a, lag=7, type=c("Box-Pierce"), fitdf=2)
```

```
##
##  Box-Pierce test
##
## data:  res_a
## X-squared = 7.0266, df = 5, p-value = 0.2187
```

```
Box.test(res_a, lag=7, type=c("Ljung-Box"), fitdf=2)
```

```
##
##  Box-Ljung test
##
## data:  res_a
## X-squared = 8.1271, df = 5, p-value = 0.1494
```

```r
Box.test((res_a)^2, lag=7, type=c("Ljung-Box"), fitdf=0)
```

```
##
##  Box-Ljung test
##
## data:  (res_a)^2
## X-squared = 2.1241, df = 7, p-value = 0.9527
```

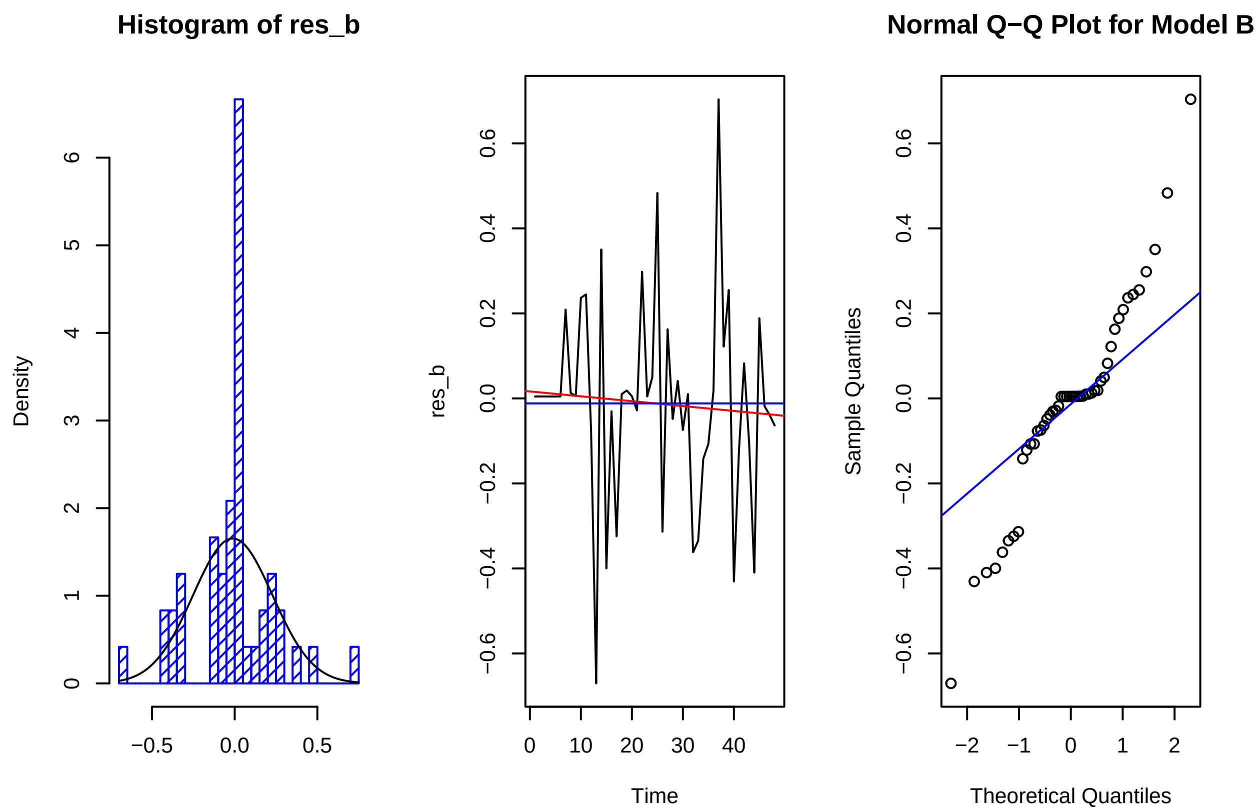
```r
ar(res_a, aic=TRUE, order.max=NULL, method=c("yule-walker"))
```

```
##
## Call:
## ar(x = res_a, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.06038
```

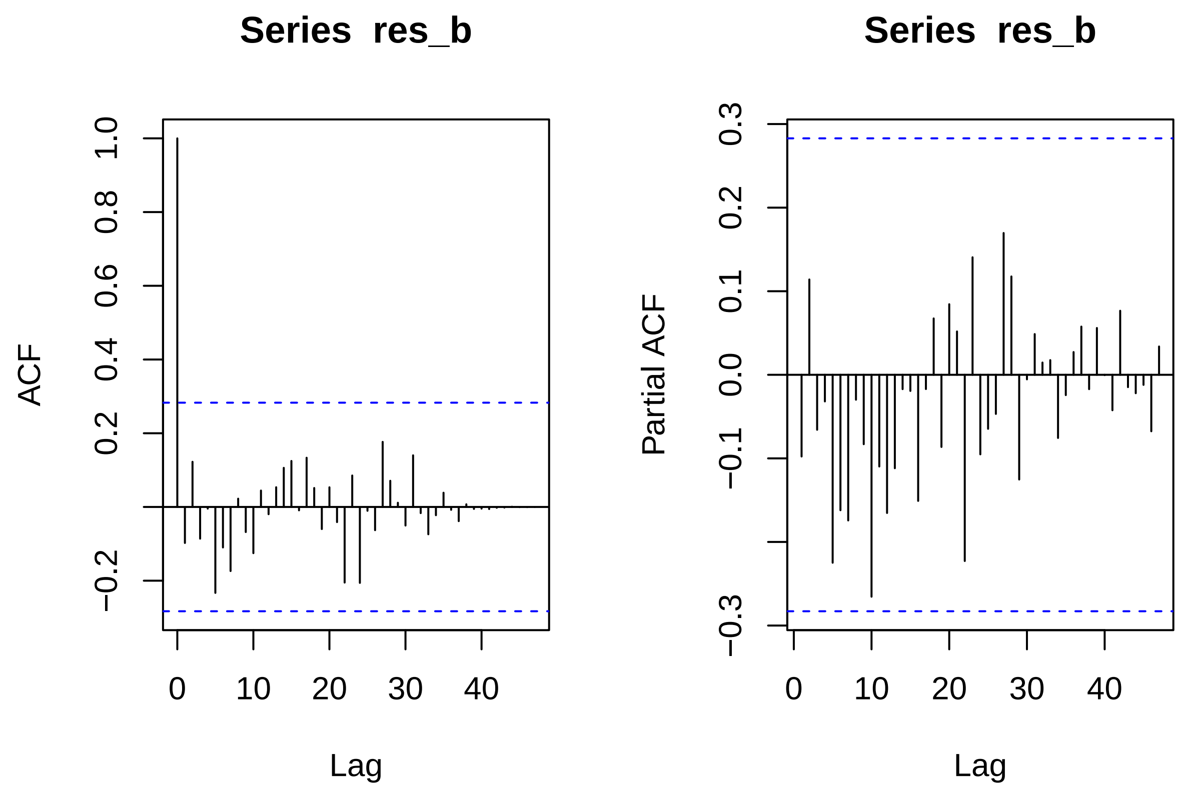
From the QQ-plot for residual of Model A, the data doesn't look like very normal, and the PACF graph for the residual have some lags that are outside of the confidence interval, so Model A may not be a good choice.

Diagnostic checking for Model B

```r
#Model B
par(mfrow=c(1,3))
res_b <- residuals(Model_B)
hist(res_b,density=20, breaks=20, col='blue', xlab='', prob=TRUE)
m_b <- mean(res_b)
std_b <- sqrt(var(res_b))
curve(dnorm(x,m_b,std_b), add=T)
plot.ts(res_b)
fitt_b <- lm(res_b~as.numeric(1:length(res_b)))
abline(fitt_b, col='red')
abline(h=mean(res_b), col='blue')
qqnorm(res_b, main="Normal Q-Q Plot for Model B")
qqline(res_b, col='blue')
```




```
par(mfrow=c(1,2))
acf(res_b, lag.max=50)
pacf(res_b, lag.max=50)
```




```
shapiro.test(res_b)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res_b
## W = 0.93859, p-value = 0.01433
```

```
Box.test(res_b, lag=7, type=c("Box-Pierce"), fitdf=2)
```

```
##
##  Box-Pierce test
##
## data:  res_b
## X-squared = 6.1747, df = 5, p-value = 0.2896
```

```
Box.test(res_b, lag=7, type=c("Ljung-Box"), fitdf=2)
```

```
##
##  Box-Ljung test
##
## data:  res_b
## X-squared = 7.1597, df = 5, p-value = 0.209
```

```r
Box.test((res_b)^2, lag=7, type=c("Ljung-Box"), fitdf=0)
```

```
##
##  Box-Ljung test
##
## data:  (res_b)^2
## X-squared = 2.5219, df = 7, p-value = 0.9254
```

```r
ar(res_b, aic=TRUE, order.max=NULL, method=c("yule-walker"))
```

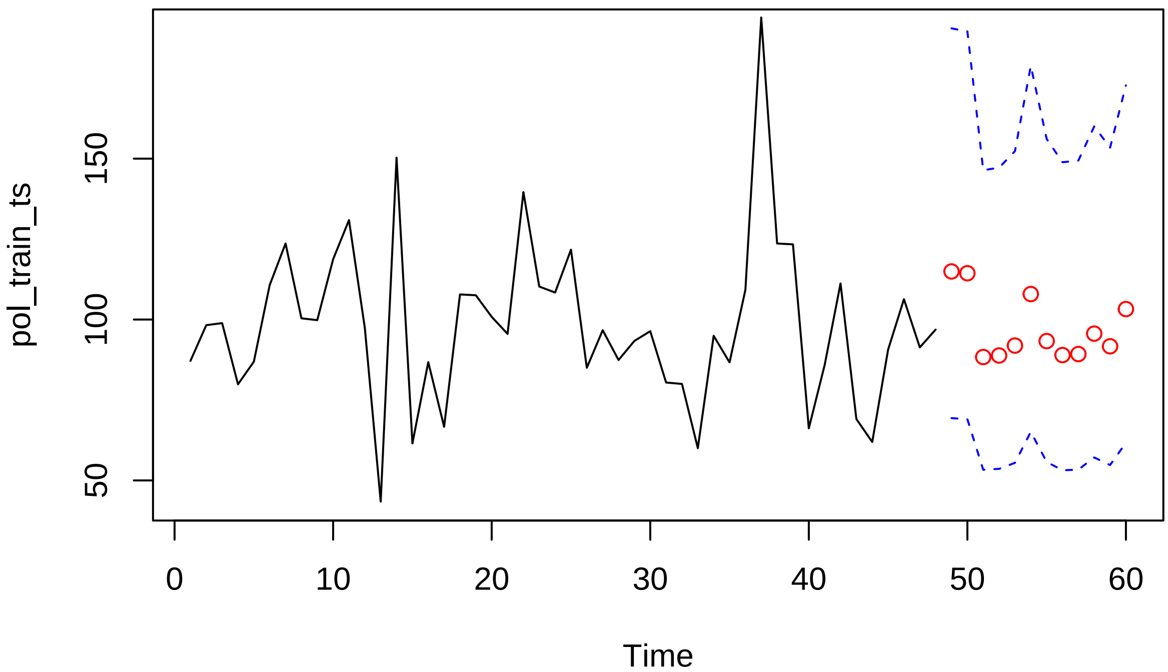
```
##
## Call:
## ar(x = res_b, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.0582
```

From the test results, although Model B didn't pass the Shapiro-Wilk normality test, but it doean't mean Model B is impossible to use. Since the residual of Model A is not White Noise, so Model B is a better choice here.

Forecasting

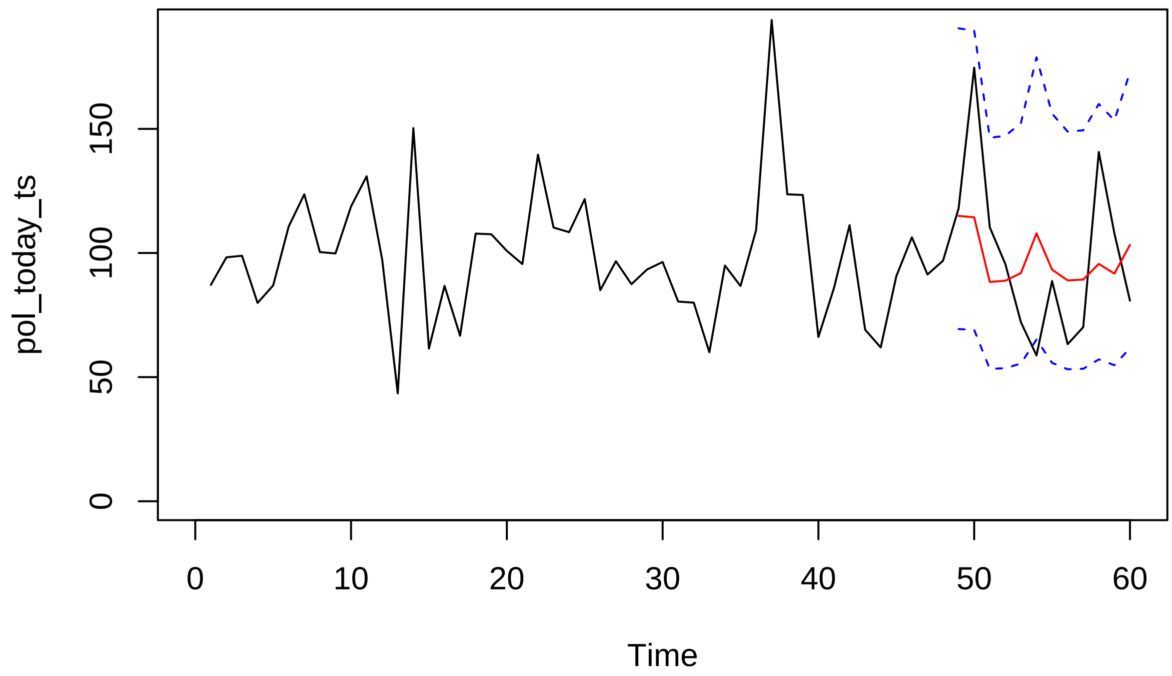
```r
pred.tr <- predict(Model_B, n.ahead = 12)
U.tr <- pred.tr$pred + 1.96 * pred.tr$se
L.tr <- pred.tr$pred - 1.96 * pred.tr$se
```

```r
pred.ori <- exp(pred.tr$pred)
U <- exp(U.tr)
L <- exp(L.tr)
ts.plot(pol_train_ts, xlim=c(1,length(pol_train_ts)+12), ylim=c(min(pol_train_ts),max(U)))
lines(U, col='blue',lty='dashed')
lines(L, col='blue',lty='dashed')
points((length(pol_train_ts)+1):(length(pol_train_ts)+12),pred.ori, col="red")
```




```
# lines((length(pol_train_ts)+1):(length(pol_train_ts)+12),pred.ori, col="red")
```

```
ts.plot(pol_today_ts, xlim=c(0, length(pol_train_ts)+12), ylim=c(0, max(U)))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
#points((length(pol_train_ts)+1):(length(pol_train_ts)+12),pred.ori, col="red")
lines((length(pol_train_ts)+1):(length(pol_train_ts)+12),pred.ori, col="red")
```




```
library(Metrics)
```

```
##
## Attaching package: 'Metrics'

## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
rmse_value <- rmse(pred.ori, pol_test_ts)
rmse_value
```

```
## [1] 30.04416
```

The MSE for Model B prediction is 30.04416.

Therefore, the final model is $SARIMA(0,0,0)(1,1,1)_6$